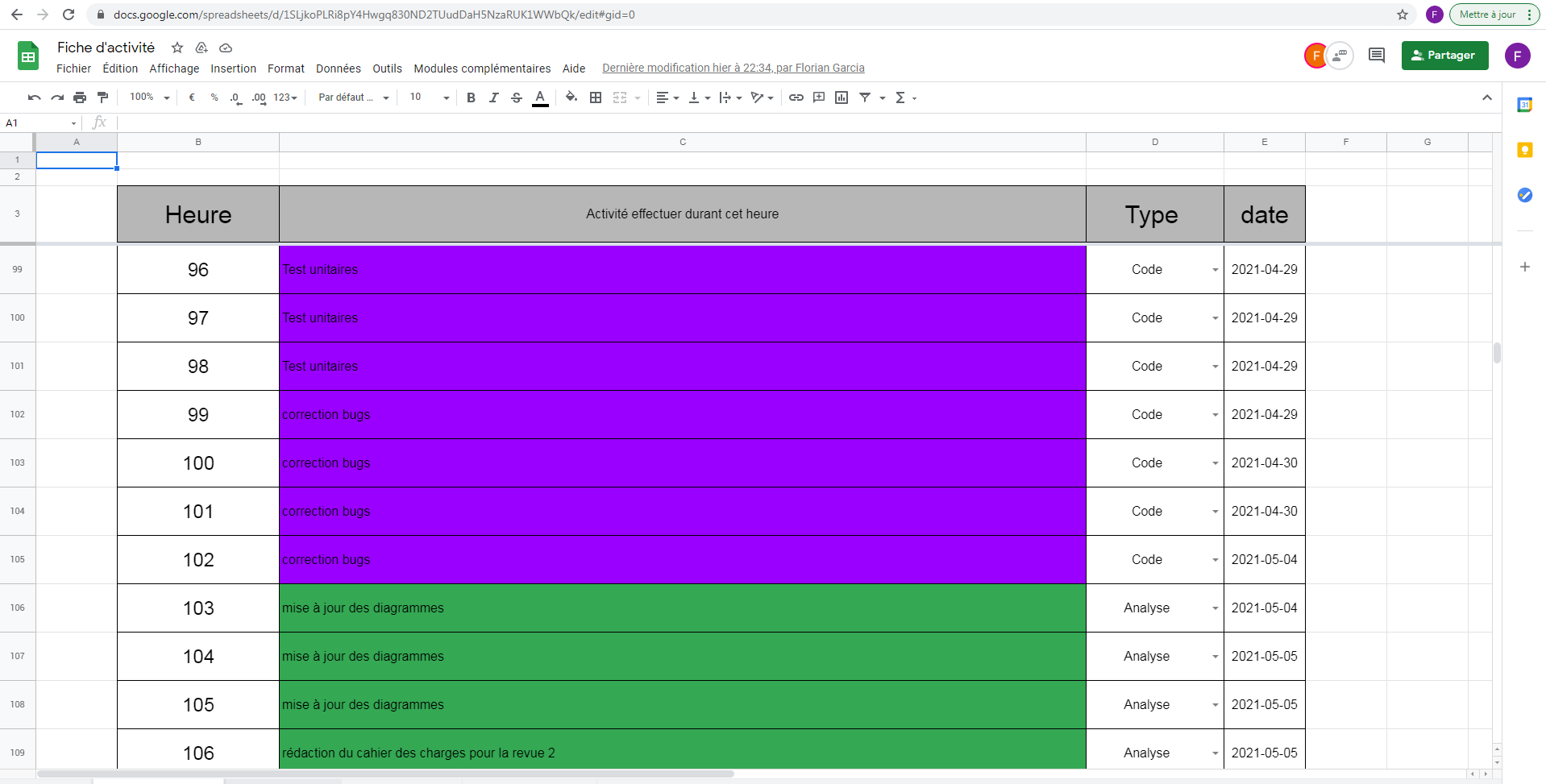
# PARTIE 2 : Partie individuel de Wantelez Florian

## Organisation

### *Compte rendu d’activité*



Le compte rendu d’activité permet un suivi horaire de chaque tâche effectuée sur le projet, qui sont détaillés en 5 types : conception, code, analyse, prototypage, matériel avec un code couleur correspondant.

#### Cahier de bord

Pour le projet a été mis en place un cahier de bord personnel permettant de suivre les différentes étapes du projet de manière plus détaillé, on peut y trouver les problèmes rencontrés, les solutions trouvées ainsi que les différentes phases de développement et mise à jour.

## Fonctionnalités du système

### *Use case*

### C:\CrousRadiateur\Revue\screen\USE_CASE_FINAL.PNG

### 

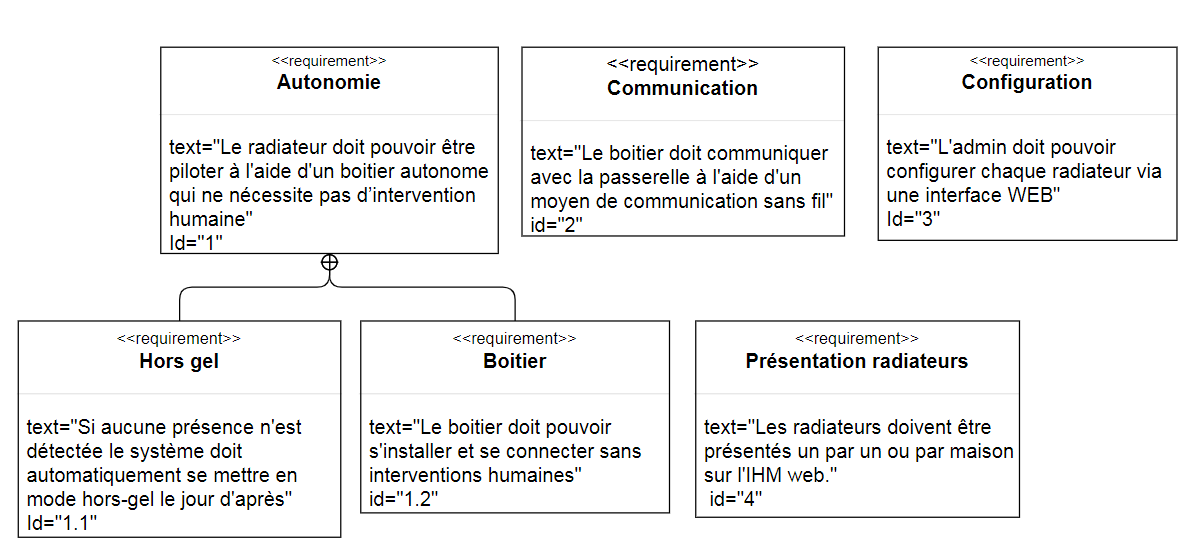
Mon diagramme de cas d’utilisation est composé de la fonctionnalité transmettre les flux et communiquer avec la passerelle, je dois pouvoir concevoir un programme permettant la réalisation de ces deux cas d’utilisations.

## *Fonctionnalité du système*

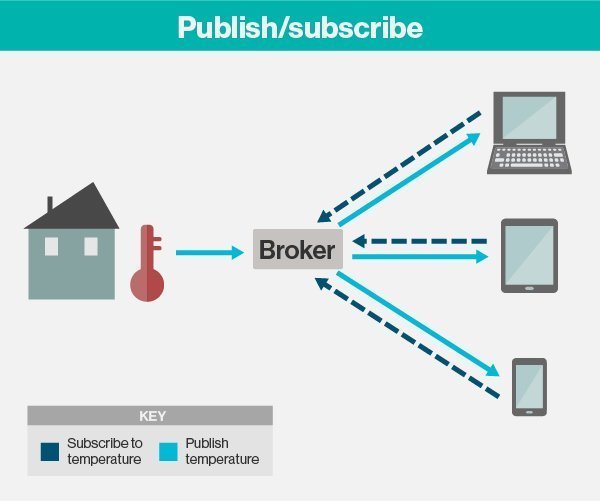
La passerelle faisant office d’intermédiaire doit être capable de transmettre les messages reçus provenant de la centrale ou du boitier autonome. La centrale quand elle doit pouvoir envoyer des messages à la passerelle une consigne de température ainsi que de recevoir les messages envoyer par la passerelle.

### *Scénario*

### *Exigence :*



### Choix du protcole



Pour développer mon application j’ai choisi d’utiliser le protocole MQTT, en effet ce protocole possède de nombreux avantages, il est très léger, flexible et fiable idéal pour envoyer des messages comportement peu de donné or c’est exactement ce que nous avons besoin de faire pour ce projet.

### Bibliothèque

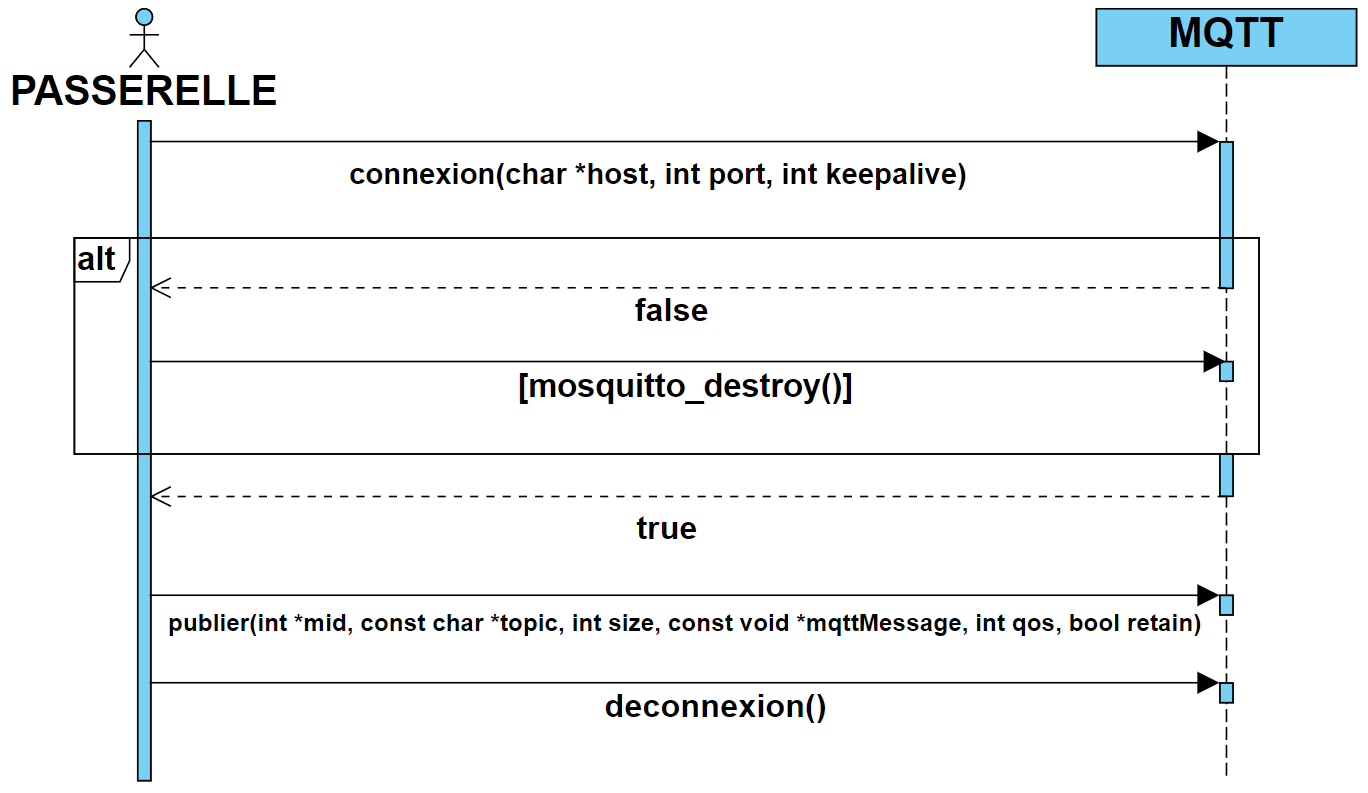
**Une image contenant texte

Description générée automatiquement**

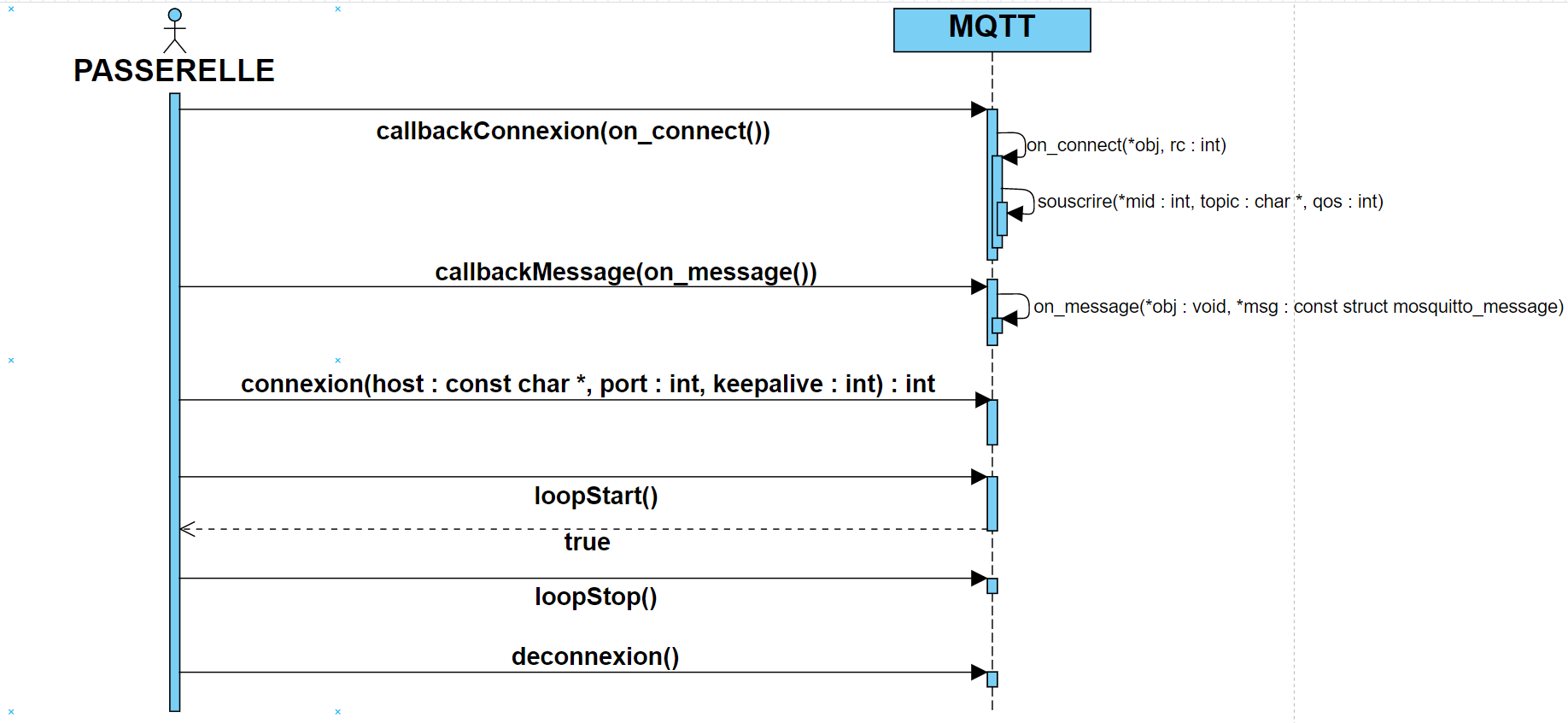
Pour développer mon application à l’aide du protocole MQTT il m’a fallu utiliser la bibliothèque mosquito, pour cela je me suis aidé de la documentation officielle disponible sur internet, dans ce document on a accès aux différentes méthodes de la bibliothèque ainsi que de leurs prototypes, j’ai pu donc utiliser les méthodes pour construire mon programme.

### *Séquence :*

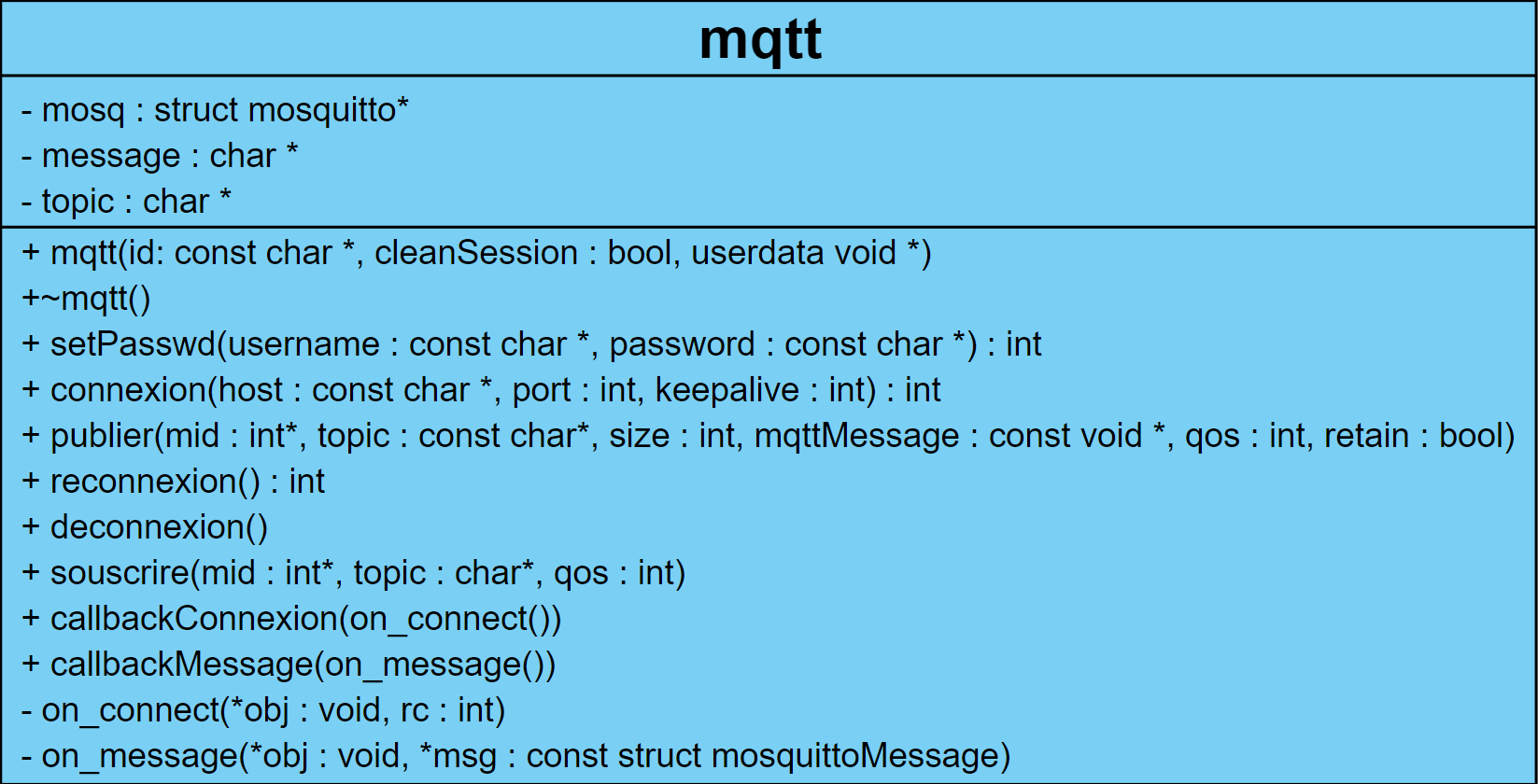
Envoyer un message à la centrale



Recevoir un message de la centrale

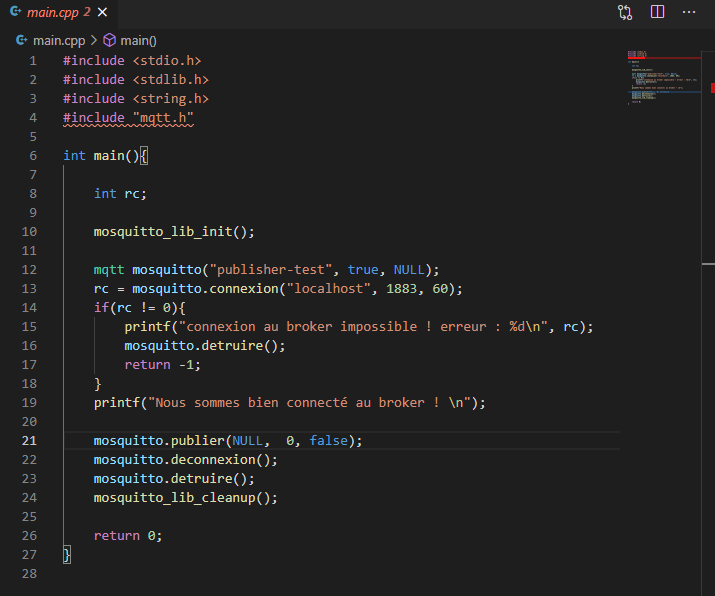


### *Classe :*

****

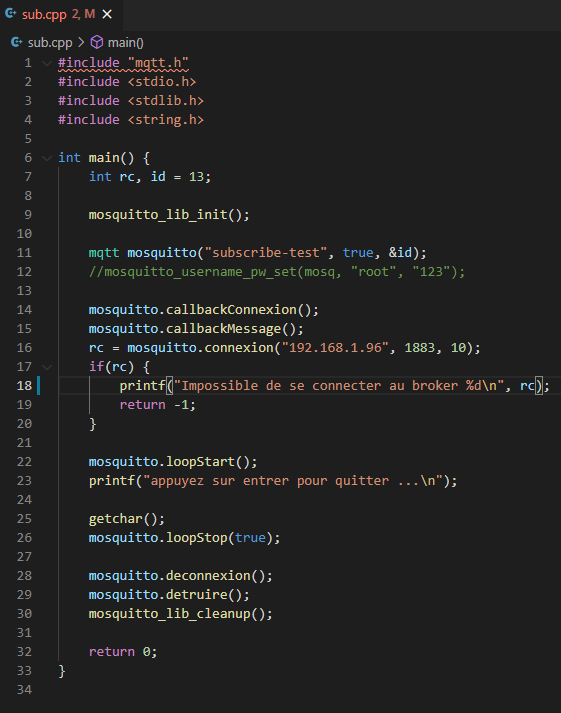
### *Code :*

Envoyer un message grâce à mosquitto :

****

Pour commencer j’ai développé le parti permettant l’envoi des messages en MQTT. Le code se décompose en plusieurs parties, tout d’abord on initialise la librairie mosquitto, puis on instancie notre objet mqtt, une fois instancié on tente une connexion au broker, si cette connexion est bien étable alors on peut publier un message sur les topics configurer, les topics sont des paramètres de la classe.

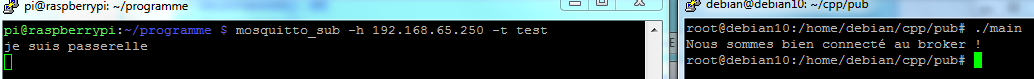
Recevoir un message grâce à mosquitto :



Cette partie permet de recevoir des messages en MQTT, tout d’abord j’appelle deux callbacks qui permettront de réagir à toute connexion ou message envoyé sur le broker, puis on se connecte au broker et une loop commence permettant d’écouter en boucle toutes les connexions et messages reçus sur le broker.

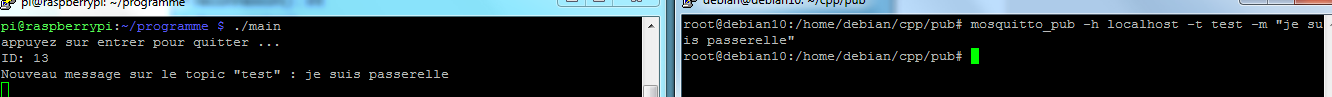
### Test :

Envoyer un message à la centrale

****

Pour ce test j’utilise la fonction mosquitto\_sub disponible dans le paquet mosquitto (screen de gauche) pour écouter les messages MQTT reçus sur le topic « test » du broker puis je lance mon programme main.cpp avec la commande ./main qui va s’occuper d’envoyer lui-même le message prédéfini, on peut voir qu’il arrive bel et bien à se connecter au broker et que le message est bien envoyé puisqu’on le reçoit dans la console de gauche.

Recevoir un message de la passerelle

****

Pour ce test j’utilise la fonction mosquitto\_pub disponible dans le paquet mosquitto (screen de gauche) pour envoyer un message en MQTT sur le topic « test » du broker puis je lance mon programme sub.cpp avec la commande ./sub qui va s’occuper d’écouter les messages reçus sur les topics configurés, ici le topic prédéfini est « test » on peut voir qu’il arrive bel et bien à recevoir le message.